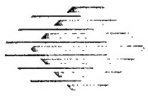


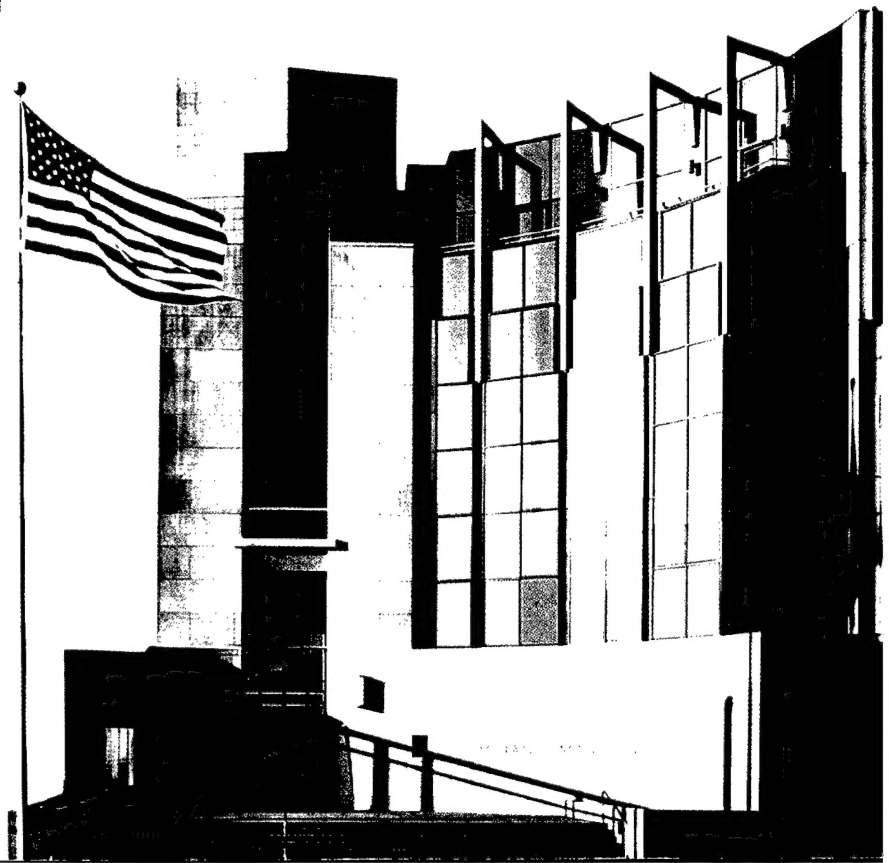
Carnegie Mellon
Software Engineering Institute

Survivable Network Analysis Method

Nancy R. Mead
Robert J. Ellison
Richard C. Linger
Thomas Longstaff
John McHugh

September 2000

TECHNICAL REPORT
CMU/SEI-2000-TR-013
ESC-2000-TR-013



Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of "Don't ask, don't tell, don't pursue" excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.

Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

Survivable Network Analysis Method

CMU/SEI-2000-TR-013
ESC-TR-2000-013

Nancy R. Mead
Robert J. Ellison
Richard C. Linger
Thomas Longstaff
John McHugh

September 2000

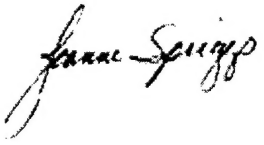
Survivable Systems

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER



Joanne E. Spriggs
Contracting Office Representative

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 52.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Abstract

Society is increasingly dependent on large-scale, networked information systems of remarkable scope and complexity. This dependency magnifies the far-reaching consequences of system damage from attacks and intrusions. Yet no amount of security can guarantee that systems will not be penetrated. Incorporating survivability capabilities into an organization's systems can mitigate the risks. Survivability is the capability of a system to fulfill its mission in a timely manner despite intrusions, failures, or accidents. The three tenets of survivability are (1) resistance to intrusions, (2) recognition of intrusion effects, and (3) recovery of services despite successful intrusions. The survivability of existing or planned systems can be analyzed at the level of system architectures or requirements. This report describes the Survivable Network Analysis (SNA) method developed at the SEI's CERT® Coordination Center. The four-step SNA method guides stakeholders through an analysis process intended to improve system survivability when a system is threatened. The method focuses on preservation of essential system services that support the organizational mission. SNA findings are summarized in a Survivability Map that enumerates current and recommended architectural strategies. SNA has been successfully applied to commercial and governmental systems, and continues to evolve toward increasing rigor in its application.

® CERT and CERT Coordination Center are registered in the U.S. Patent and Trademark Office.

1 Introduction

Today's large-scale, highly distributed, networked systems improve the efficiency and effectiveness of organizations by permitting whole new levels of organizational integration. However, such integration is accompanied by elevated risks of intrusion and compromise. Incorporating survivability capabilities into an organization's systems can mitigate these risks.

As an emerging discipline, survivability builds on related fields of study (e.g., security, fault tolerance, safety, reliability, reuse, performance, verification, and testing) and introduces new concepts and principles. Survivability focuses on preserving essential services, even when systems are penetrated and compromised [Anderson 97].

Current software-development life-cycle models are not focused on creating survivable systems, and exhibit shortcomings when used to develop systems that are intended to have a high degree of assurance of survivability [Marmor-Squires 88]. If addressed at all, survivability issues are often relegated to a separate thread of project activity, with the result that survivability is treated as an add-on property. This isolation of survivability considerations from primary system-development tasks results in an unfortunate separation of concerns. Survivability should be integrated and treated on a par with other system properties in order to develop systems that have the required functionality and performance, but can also withstand failures and compromises. Important design decisions and tradeoffs become more difficult when survivability is not integrated into the primary development life cycle. Separate threads of activities are expensive and labor intensive, often duplicating effort in design and documentation. In addition, tools for supporting survivability engineering are often not integrated into the software-development environment. With separate threads of activities, it becomes more difficult to adequately address the high-risk issues of survivability and the consequences of failure. In addition, technologies that support survivability goals, such as formal specification, architecture tradeoff methods, intrusion analysis, and survivability design patterns, are not effectively applied into the development process.

For each life-cycle activity, survivability goals should be addressed, and methods to ensure survivability incorporated. In some cases, existing development methods can enhance survivability. Current research is creating new methods that can be applied; however, more research and experimentation are required before the goal of survivability can become a reality. In this paper, we describe survivability concepts, discuss a software-development life-cycle model for survivability, and describe a technique that can be applied during requirements, specification, and architecture activities to support survivability goals.

2 Definition of Survivability

2.1 Survivability Concepts

Survivable systems research over the past few years has resulted in development of the concepts and definitions of survivability described in this section, which are drawn from the work of the Survivable Network Technology team at the Software Engineering Institute (SEI) and the CERT® Coordination Center (CERT/CC) [Ellison 99].

2.1.1 The New Network Paradigm: Organizational Integration

From their modest beginnings some 20 years ago, computer networks have become a critical element of modern society. These networks not only have global reach, they also affect virtually every aspect of human endeavor. Network systems are principal enabling agents in business, industry, government, and defense. Major economic sectors, including defense, energy, transportation, telecommunications, manufacturing, financial services, health care, and education, all depend on a vast array of networks operating on local, national, and global scales. This pervasive societal dependency on networks magnifies the consequences of intrusions, accidents, and failures, and amplifies the critical importance of ensuring network survivability.

As organizations seek to improve efficiency and competitiveness, a new network paradigm is emerging. Networks are being used to achieve radical new levels of organizational integration. This integration obliterates traditional organizational boundaries and transforms local operations into components of comprehensive, network-resident business processes. For example, commercial organizations are integrating operations with business units, suppliers, and customers through large-scale networks that enhance communication and services. These networks combine previously fragmented operations into coherent processes open to many organizational participants. This new paradigm represents a shift from bounded networks with central control to unbounded networks. Unbounded networks are characterized by distributed administrative control without central authority, limited visibility beyond the boundaries of local administration, and lack of complete information about the network. At the same time, organizational dependencies on networks are increasing and the risks and consequences of intrusions and compromises are amplified.

® CERT and CERT Coordination Center are registered in the U.S. Patent and Trademark Office.

2.1.2 The Definition of Survivability

We define *survivability* as the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents. We use the term *system* in the broadest possible sense, including networks and large-scale systems of systems.

The term *mission* refers to a set of very high-level requirements or goals. Missions are not limited to military settings, because any successful organization or project must have a vision of its objectives whether expressed implicitly or as a formal mission statement. Judgments as to whether or not a mission has been successfully fulfilled are typically made in the context of external conditions that may affect achievement of that mission. For example, imagine that a financial system shuts down for 12 hours during a period of widespread power outages caused by a hurricane. If the system preserves the integrity and confidentiality of its data and resumes its essential services after the period of environmental stress is over, the system can reasonably be judged to have fulfilled its mission. However, if the same system shuts down unexpectedly for 12 hours under normal conditions or minor environmental stress, thereby depriving its users of essential financial services, the system can reasonably be judged to have failed its mission even if data integrity and confidentiality are preserved.

Timeliness is a critical factor that is typically included in (or implied by) the very high-level requirements that define a mission. However, timeliness is such an important factor that we included it explicitly in the definition of survivability.

The terms *attack*, *failure*, and *accident* are meant to include all potentially damaging events; but in using these terms we do not partition these events into mutually exclusive or even distinguishable sets. It is often difficult to determine if a particular detrimental event is the result of a malicious attack, a failure of a component, or an accident. Even if the cause is eventually determined, the critical immediate response cannot depend on such speculative future knowledge.

Attacks are potentially damaging events orchestrated by an intelligent adversary. Attacks include intrusions, probes, and denials of service. Moreover, the threat of an attack may have as severe an impact on a system as an actual occurrence. A system that assumes a defensive position because of the threat of an attack may reduce its functionality and divert additional resources to monitor the environment and protect system assets.

We include failures and accidents in the definition of survivability. *Failures* are potentially damaging events caused by deficiencies in the system or in an external element on which the system depends. Failures may be due to software design errors, hardware degradation, human errors, or corrupted data. The term *accidents* comprises a broad range of randomly occurring and potentially damaging events such as natural disasters. We tend to think of accidents as externally generated events (i.e., outside the system) and failures as internally generated events.

With respect to system survivability, a distinction between a failure and an accident is less important than the impact of the event. Nor is it often possible to distinguish between intelligently orchestrated attacks and unintentional or randomly occurring detrimental events. Our approach concentrates on the effect of a potentially damaging event. Typically, for a system to survive, it must react to and recover from a damaging effect (e.g., the integrity of a database is compromised) long before the underlying cause is identified. In fact, the reaction and recovery must be successful whether or not the cause is ever determined.

The primary focus in this paper is to provide managers with methods to help systems survive the acts of intelligent adversaries. While the focus is on intrusions, the methods discussed apply in full measure to failures and accidents as well.

Finally, it is important to recognize that it is the mission fulfillment that must survive, not any particular subsystem or system component. Central to the notion of survivability is the capability of a system to fulfill its mission, even if significant portions of the system are damaged or destroyed. We use the term *survivable system* as a shorthand for a system with the capability to fulfill a specified mission in the face of attacks, failures, or accidents. Again, it is the mission, not a particular portion of the system, that must survive.

2.1.2.1 Characteristics of Survivable Systems

A key characteristic of survivable systems is their capability to deliver essential services in the face of attack, failure, or accident. Central to the delivery of essential services is the capability of a system to maintain essential properties (i.e., specified levels of integrity, confidentiality, performance, and other quality attributes) in adverse environments. Thus, it is important to define minimum levels of such quality attributes that must be associated with essential services. For example, a launch of a missile by a defensive system is no longer effective if the system performance is slowed to the point that the target is out of range before the system can launch.

These quality attributes are so important that definitions of survivability are often expressed in terms of maintaining a balance among multiple quality attributes, such as performance, security, reliability, availability, fault tolerance, modifiability, and affordability. The Architecture Tradeoff Analysis project at the Software Engineering Institute is using this attribute-balancing (i.e., tradeoff) view of survivability to evaluate and synthesize survivable systems [Kazman 98]. Quality attributes represent broad categories of related requirements, so a quality attribute may be composed of other quality attributes. For example, the security attribute traditionally includes three sub-attributes, namely, confidentiality, integrity, and availability.

The capability to deliver essential services and maintain associated essential properties must be sustained even if a significant portion of a system is incapacitated. Furthermore, this capability should not be dependent upon the survival of a specific information resource, computation, or communication link. In a military setting, essential services might be those required

to maintain an overwhelming technical superiority, and essential properties may include integrity, confidentiality, and a level of performance sufficient to deliver results in less than one decision cycle of the enemy. In the public sector, a survivable financial system might be one that maintains the integrity, confidentiality, and availability of essential information and financial services, even if particular nodes or communication links are incapacitated through intrusion or accident, and that recovers compromised information and services in a timely manner. The financial system's survivability might be judged by using a composite measure of the disruption of stock trades or bank transactions (i.e., a measure of the disruption of essential services).

Key to the concept of survivability, then, is identifying the essential services (and the essential properties that support them) within an operational system. Essential services are defined as the functions of the system that must be maintained when the environment is hostile or failures or accidents occur that threaten the system. To maintain their capabilities to deliver essential services, survivable systems must exhibit the four key properties illustrated in Table 1, namely resistance, recognition, recovery (the three R's), and adaptation.

Table 1: Properties of Survivable Systems

Key Property	Description	Example Strategies
Resistance to attacks	Strategies for repelling attacks	Authentication Access controls Encryption Message filtering Survivability wrappers System diversification Functional isolation
Recognition of attacks and damage	Strategies for detecting attacks and evaluating damage	Intrusion detection Integrity checking
Recovery of essential and full services after attack	Strategies for limiting damage, restoring compromised information or functionality, maintaining or restoring essential services within mission time constraints, restoring full services	Redundant components Data replication System backup and restoration Contingency planning
Adaptation and evolution to reduce effectiveness of future attacks	Strategies for improving system survivability based on knowledge gained from intrusions	New intrusion recognition patterns

The table identifies a number of survivability strategies that can be applied to counter threats of an overt attack on a system. Some of these techniques for enhancing survivability are borrowed from other areas, notably the security, safety, and fault-tolerance communities.

In the area of attack resistance, a number of techniques are available. User-authentication mechanisms limit access to a system to a group of approved users. Authentication mechanisms range from simple passwords to combinations of passwords, user-carried authentication tokens (themselves password protected), and biometrics. Access controls can be applied to system access or to individual programs and data sets. Access controls, enforced by a trustworthy operating system, automatically apply a predefined policy to grant or deny access to an authenticated user. When properly used and implemented, access controls can serve as a substitute for program- and data-set-level password mechanisms.

Encryption can protect data; either within a system or in transit between systems, from interception or physical capture. Available encryption technologies are strong enough to resist all currently feasible brute-force attacks. Encryption translates the problem of protecting large quantities of data into a problem of managing relatively small quantities of keying material. Encryption can also be used to provide authentication, non-repudiation, integrity checking, and a variety of other assurance properties.

Message filtering is typically used at the boundary of a system or installation to restrict the traffic that enters the system. For example, there is no reason to allow messages related to unsupported or unwanted services to enter an installation. Messages appearing to originate from within an installation are probably not legitimate if coming from the outside. Such messages should not be let out. Filters can be designed to block messages associated with known attacks, as well.

Survivability wrappers are essentially message filters applied at the OS interface level. They may be used to provide operand checking or to redirect calls to unsafe library routines to more robust versions. They may also be used to impose a restrictive access-control policy on a particular application. System diversification combined with redundant implementations makes an attacker's job more difficult. In a diverse implementation, it is likely that a scenario used to attack one implementation will fail on others. Defensive coding is used to protect programs from bad input values. Functional isolation reduces or eliminates dependencies among services to the greatest extent possible. This prevents an attack on one service from compromising others. Isolation is often not easy to achieve, as dependencies among services are not obvious if viewed at the wrong level of abstraction. Services that share a processor, for example, are mutually dependent on one another for CPU and memory resources. They may also share disk space and probably a network adapter. It is possible for one process to launch a denial-of-service attack on another by gaining a monopoly on any of these resources. Resource isolation may require a quota-based sharing mechanism or similar technique. Functional isolation can extend to physically separating system functions, often on separate servers with no logical connections—for example, separating email processing from sensitive data files. No electronic intrusion method can jump an air gap or penetrate a machine that is powered down.

In the area of attack recognition, there are a limited number of choices. Intrusion-detection systems typically attempt to identify attacks by either looking for evidence of known attack patterns or by using a baseline model of normal system behavior to treat departures from normal as potential attacks. Both techniques can be applied to network traffic as well as to platform- or application-specific data. System auditing and application logs are sources of information for detecting intrusions at the platform or application level. Both real-time and post-processing intrusion-detection systems exist. At the present time, these systems miss many intrusions, especially new or novel attacks, and suffer from high false-alarm rates. Integrity checkers can detect intrusions that modify system files or data that should remain unchanged. The checking process involves creating a baseline model of the files to be protected using checksums or cryptographic signatures, and periodically comparing the current model to the baseline.

In terms of recovery, when a damaging attack (or other failure) is recognized, it is necessary to take steps to immediately recover essential services and, eventually, full services. There are a number of techniques that can be used. Their effects range from transparent maintenance of full services without noticeable interruption to fallback positions that maintain only a core of essential services.

Redundancy is the key to maintaining full services in the face of failures. The fault-tolerance community has considerable experience in the use of redundancy to maintain service in the face of component failures, but their analytical techniques are predicated on knowing the statistical distributions associated with various failure mechanisms, something that may not be possible with failures induced by attacks.

In many cases, the replication of critical data is a primary means for achieving recovery. When essential services are supplied through commodity databases or the like, it may be possible to restore a critical data service by simply starting up another instance of the commodity server with a replicated database at a more or less arbitrary location.

Systematic backup of all data sources, combined with appropriate mechanisms for restoring the data on originating or other platforms, is a key part of any recovery strategy. The granularity of backups should depend on the frequency at which data changes and the cost of repeating the work performed between backups. In extreme cases, it may be necessary to back up files each time they are closed after writing, and to log transactions or keystrokes so that intermediate work can be recovered. In other cases, daily or weekly backups may suffice.

When a system is under attack or has experienced a failure, it may be possible to dynamically reconfigure the system to transfer essential services from the attacked component to an operational one, eliminating less essential services in the process. This strategy is employed by the Federal Reserve, which can tolerate limited outages at one of its three primary computational centers in this fashion. Because this strategy does not have redundant capacity, the re-

configuration can persist only for limited periods, as the criticality of less essential services increases with the length of time that they are unavailable.

Finally, it may be possible to devolve the system to an alternate mode of operation, perhaps one in which the role of the computer system is temporarily reduced or even eliminated. For example, computer-to-computer transactions might be replaced with manually initiated faxes. A computerized parts inventory and order system might revert for a short period to a previous manual system that indicated reorder levels by red lines on the parts storage bins.

Perhaps the hardest part of survivability is adapting a system to make it more robust in the hope that it will resist never-before-seen attacks or intrusions. Just as attackers are constantly looking for new points of vulnerability, defenders must create defenses that are based on generalizations about previously seen attacks, in an effort to anticipate the directions from which new attacks might occur.

3 Survivability Life Cycle Definition

3.1 The Spiral Model

Here we describe a life-cycle model that was developed for use in trusted systems [Marmor-Squires 89]. Such a model is a natural fit for development of survivable systems. This work was based on an assessment of the waterfall and spiral models, and an extension of the spiral model to incorporate concepts of trusted systems.

An analysis of life-cycle model work done to date led to widespread use of the TRW spiral model as a foundation. The spiral model is specialized for use in developing survivable systems. The spiral model for the software-development process has been developed at TRW as an alternative to more conventional (largely waterfall-style) models. The spiral model's key features are risk management, robustness, and flexibility. This section is devoted to a description of the basic spiral model and a specialization of it. The description of the spiral model largely follows that of Boehm [Boehm 89]. Much of the initial work on spiral models was carried out by Mills and his associates [Mills 86].

The development of software is, at best, a difficult process. Many software systems, especially in the commercial area, simply evolve over time without a well-defined development process. Other systems are developed using (or at least giving lip service to) a stagewise progression of steps, possibly with feedback between adjacent steps—for example, in the waterfall model [Royce 87]. As Parnas has pointed out, this model makes a much better ex post facto explanation of the development process than a guide for its execution [Parnas 86].

Over the years, numerous variations on, or alternatives to, the waterfall model have been proposed. Each of these alternatives overcomes certain defects in the waterfall model, but introduces its own set of problems.

While the waterfall model serves a useful purpose in introducing discipline into the software-development process, it essentially dictates the linear progressions that were necessary in the batch-oriented world of limited alternatives and scarce computational power. It assumes a factory-like, assembly-line system for which we understand each piece. At the present time, the availability of workstations, networks, and inexpensive mass storage, along with a variety of tools, makes possible a wide variety of exploratory programming activities as part of the development process. This means that it is possible to develop prototypes or models for parts of systems, obtain reactions from a potential user community, and feed this information back into the development process. Standardization efforts have produced large libraries of components and even entire subsystems that can be used to reduce the amount of new develop-

ment required for a project. The growing availability of development and execution environments has accelerated this trend.

The spiral model is an attempt to provide a disciplined framework for software development that both overcomes deficiencies in the waterfall model, and accommodates activities such as prototyping, reuse, and automatic coding as a part of the process. A consequence of the flexibility of the life-cycle model is that the developer is faced with choices at many stages of the process. With choice comes risk; therefore, much of the emphasis of the spiral model is placed on risk management. This, in turn, may result in uneven progress in various aspects of system development, with high-risk areas being explored in depth while low-risk areas are deferred.

The spiral model views the development process in polar coordinates. The *r* coordinate represents cumulative project cost and the *w* coordinate represents progress to date. The plane is divided into four quadrants that represent different kinds of activities, as follows:

I. Determination of objectives, alternatives, and constraints

II. Evaluation of alternatives; identification and resolution of risks

III. Development activities

IV. Review and planning for future cycles

In addition, the boundary between quadrants I and IV represents a commitment to move forward with a particular element, approach, or method, and advance to the next stage (or spiral) within a defined space of activities (e.g., design). Specific activities may overlap multiple spirals. Also, concurrent spirals may be required to address varying areas of risk. The commitment line may involve a decision to terminate the project or change direction based on the review results.

Figure 1 shows a single cycle of the spiral. The paragraphs that follow characterize the activities that take place in each quadrant. Note that *w* does not progress evenly with time. Some cycles of the spiral may require months to complete, while others require only days. Similarly, although increasing *w* denotes progress within a cycle of the spiral, it does not necessarily denote progress toward project completion. Each cycle of the model addresses all the activities between review and commitment events. Early in the process, cycles may be short as alternatives in the decision space of the project are explored. As risks are resolved, cycles may stretch, with the development quadrant subsuming several steps in the waterfall. The spiral may be terminated with product delivery—in which case modification or maintenance activities are new spirals—or continue until the product is retired.

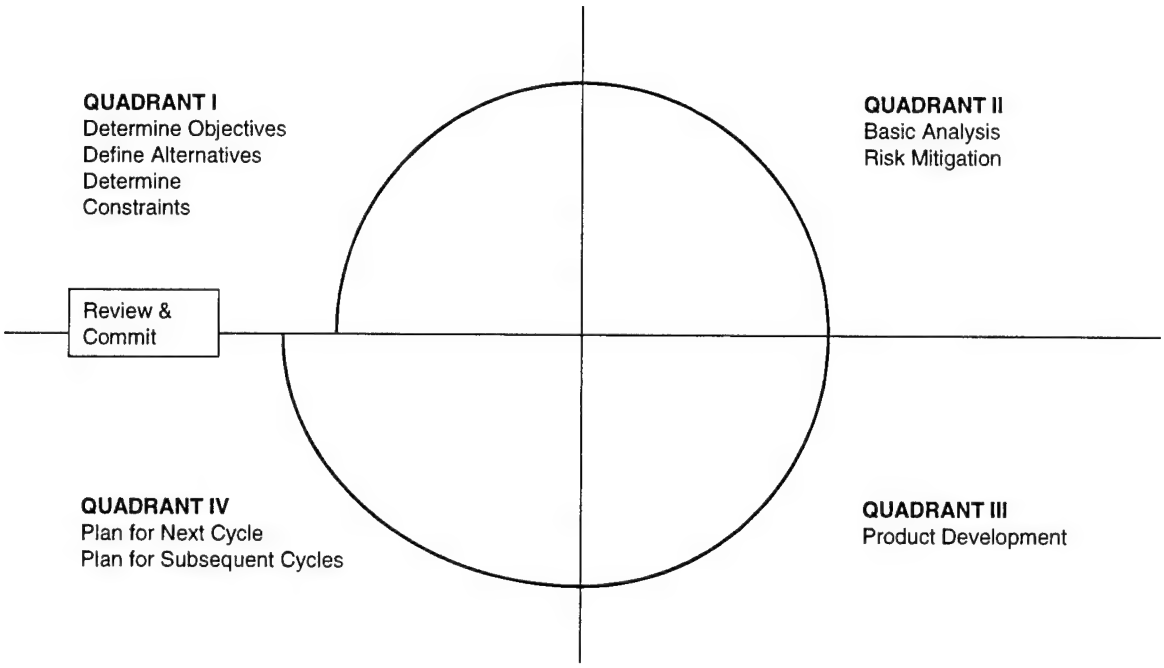


Figure 1: A Project Spiral Cycle

3.2 A Spiral Model for Survivable Systems Development

The generalized "pure" spiral process discussed above provides a framework for more specialized models. Specialization and enhancement consists of adapting the activities carried out under the model to the special requirements of the systems to be produced. This is done by specifying (1) activities that address the drivers that characterize the system, and (2) constraints that characterize the environment in which the system is to be produced.

The primary driver in the present context is the requirement to develop survivable systems. Constraints include the political and social environment in which the system is to be constructed, the ever-present cost considerations, and the limitations of technologies and knowledge that can be brought to bear on the problem at hand. These combine to yield a specialized version of the spiral model that integrates survivability into the management process, as depicted in Figure 2.

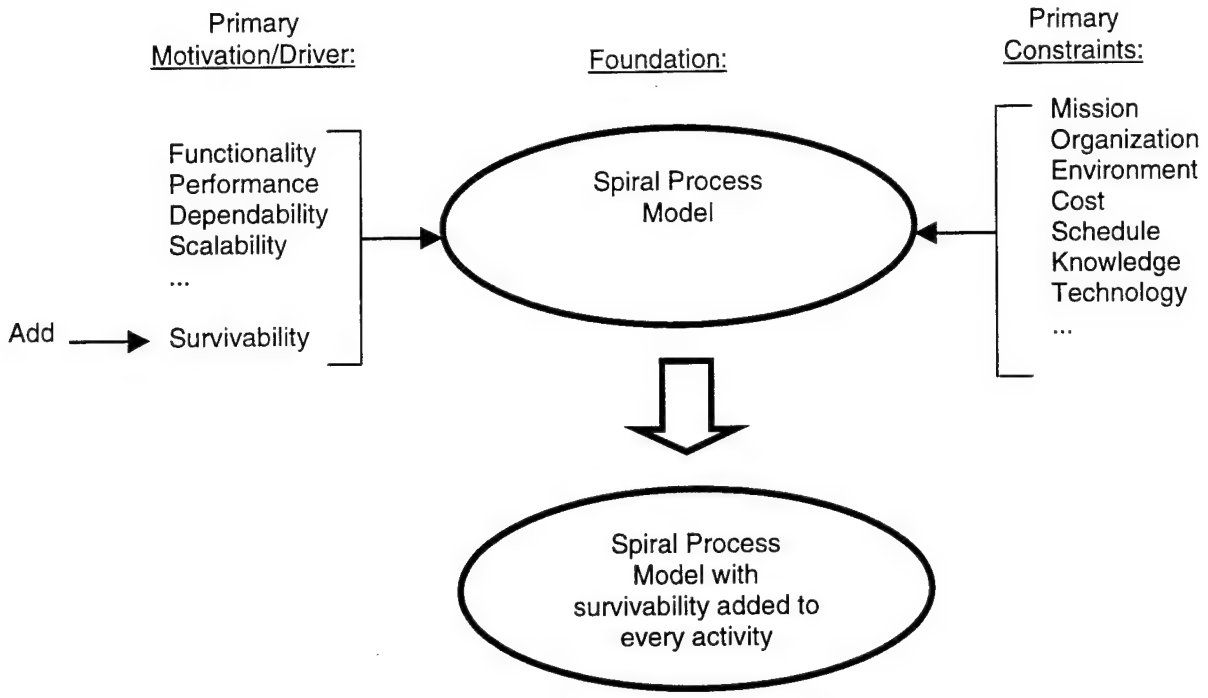


Figure 2: Specialization of the Spiral Model for Survivability Driver

Survivable systems must satisfy a variety of conflicting interests. End users want systems to carry out their primary operational mission, possibly at the expense of violating security policies under some circumstances. It is often the case that systems must also satisfy some certification or accreditation authority. The steps required for these approvals may conflict with the interests of users. Developers want to finish the job, preferably ahead of schedule and under budget. Within the development organization, tensions may exist between the various specialties involved. Resolving these conflicts may involve constraining the environment and the development process. In addition, cost considerations are always present. The spiral development process has proven to be more cost effective than traditional methods, but exhibits a different distribution of cost with time. Under the spiral model, expenditures are typically higher in early specification and design activities, resulting in cost savings in later implementation and integration activities.

Table 2 identifies a typical set of broad system-development activities and the corresponding survivability elements of each. The key point is that survivability is integrated into the broader activities. For example, in defining system requirements, the following must be defined along with survivability attributes: function, performance, dependability, scalability, and other properties. The activities in Table 2 comprise the subject matter for project management under the specialized spiral model of Figure 2.

As an illustration, consider the following imagined application of the spiral management process to the architecture-definition phase. We assume that prior phases have been completed successfully and that the appropriate requirements and specification documents are at hand. The task of the initial architecture-definition spiral is to define a set of candidate components and their interconnections that will implement the specified services in a way that satisfies both functional and non-functional requirements. The architect will choose candidate platforms, allocate functions to them, and determine the appropriate connections among platforms and between platforms and the outside world. A variety of tools and techniques will be used to analyze the proposed architecture to determine whether it satisfies the requirements and specifications. One possibility of this analysis is that the proposed architecture satisfies the functional requirements but cannot achieve the required throughput. Although processor replication has already been used to improve performance, the processors require close coupling to maintain synchronization, and their co-location presents a vulnerability as a potential single site of failure. Another spiral over the architecture is in order as unresolved risks remain.

An examination of the specification for the service that results in the bottleneck shows that what appeared as a monolithic service at first glance actually decomposes in a way that reduces the processing load and allows the two parts of the service to be separated both physically and temporally. After confirming that this revised service specification satisfies the requirements and is consistent with the other, unchanged specifications, the architecture is revisited. The revised specification permits a reduction in processor load and allows the critical function to be performed at several distant locations with greatly relaxed data-synchronization requirements. As a result, it is possible to configure the system with sufficient redundancy so that at least two loss-of-site events can be tolerated without loss of service. Further site loss will reduce service levels, but it is possible to prioritize requests so that the minimum essential service level will be maintained. Detailed analyses of this approach show a low probability race condition that could deadlock the system. Adding explicit synchronization mechanisms (another iteration) and additional communications capacity reduces the residual risk to an acceptable level, and the architecture phase is complete after two spirals of the management process.

3.3 Life-Cycle Activities and Survivability

The key survivability elements of Table 2 are the principal tasks that must be managed within the spiral model to achieve system survivability. The Survivable Network Analysis (SNA) technique has proven to be useful in requirements definition, system specification, and system architecture activities. It will be described in this report.

Table 2: Life-Cycle Activities and Corresponding Survivability Elements

Life-Cycle Activities	Key Survivability Elements	Examples
Mission Definition	Analysis of mission criticality and consequences of failure	Estimation of cost impact of denial of service attacks
Concept of Operations	Definition of system capabilities in adverse environments	Enumeration of critical mission functions that must withstand attacks
Project Planning	Integration of survivability into life-cycle activities	Identification of defensive coding techniques for implementation
Requirements Definition	Definition of survivability requirements from mission perspective	Definition of access requirements for critical system assets during attacks
System Specification	Specification of essential service and intrusion scenarios	Definition of steps that compose critical system transactions
System Architecture	Integration of survivability strategies into architecture definition	Creation of network facilities for replication of critical data assets
System Design	Development and verification of survivability strategies	Correctness verification of data encryption algorithms
System Implementation	Application of survivability coding and implementation techniques	Definition of methods to avoid buffer overflow vulnerabilities
System Testing	Treatment of intruders as users in testing and certification	Addition of intrusion usage to usage models for statistical testing
System Evolution	Improvement of survivability to prevent degradation over time	Redefinition of architecture in response to changing threat environment

4 Survivable Network Analysis Steps

The SNA method permits systematic assessment of the survivability properties of proposed systems, existing systems, and modifications to existing systems. The analysis is carried out at the architecture level as a cooperative project by a customer team and an SEI team. The method proceeds through a series of joint working sessions, culminating in a briefing on findings and recommendations. Figure 3 depicts the four-step SNA process.

In Step 1, System Definition, the business mission of the system and its primary functional requirements are elicited. The usage environment is discussed in terms of the capabilities and locations of system users, and the types and volumes of system transactions. System risks are reviewed in terms of the types of adverse conditions that may be encountered. The system architecture is elicited in terms of hardware components and connections, software configurations, and information residency.

In Step 2, Essential Capability Definition, the essential services and assets of the system are selected. Essential services and assets are those capabilities critical to fulfilling the business mission of the system, and which must be maintained under adverse conditions. The essential service usage scenarios that invoke essential services and access essential assets are then defined. Usage scenarios are composed of the successive steps required for users to invoke the services and access the assets. Finally, the usage scenarios are traced through the system architecture to identify the essential components that participate in providing the essential services and assets. This tracing process amounts to mental execution of the scenarios, as they traverse successive components in the architecture.

In Step 3, Compromisable Capability Definition, a set of representative intrusions is selected based on the system's operating environment. Intrusion usage scenarios are defined and traced through the architecture to identify compromisable components that the intrusions could successfully access and damage.

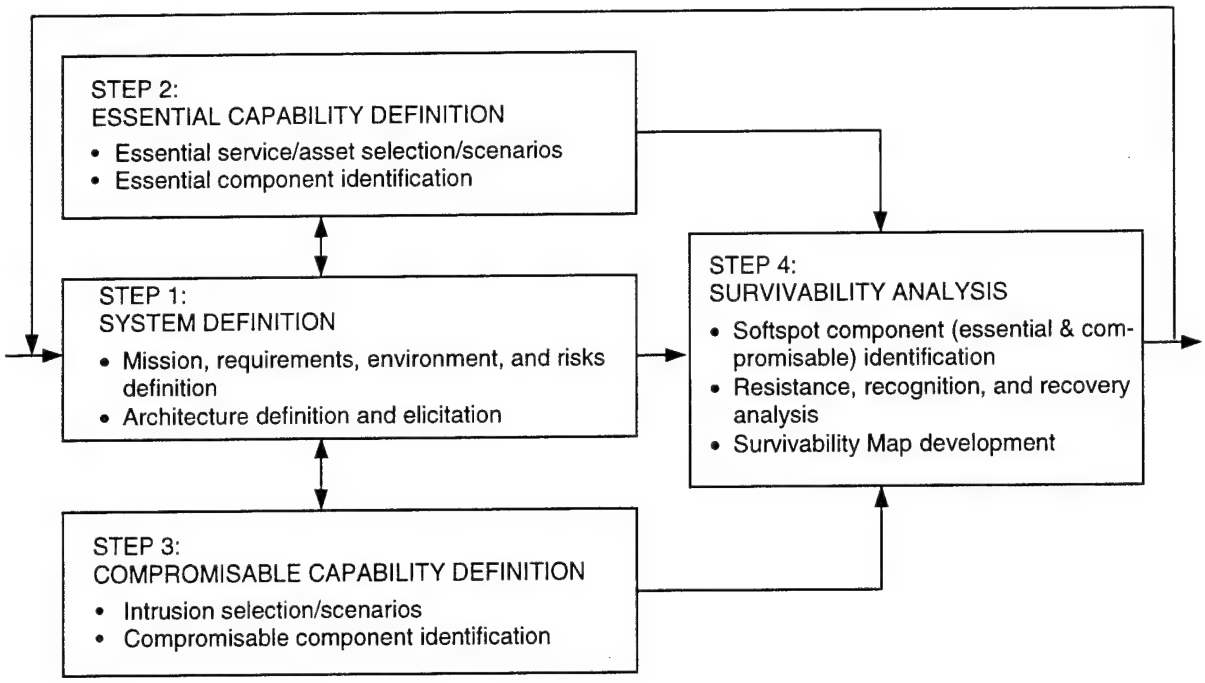


Figure 3: The Survivable Network Analysis Method

In Step 4, Survivability Analysis, softspot components are identified as those components that are both essential and compromisable. The architecture is then analyzed for softspot protection, in terms of its capability to resist, recognize, and recover from intrusions. Architectural recommendations are then formulated and summarized in a Survivability Map, as illustrated in Figure 4 [Ellison 98]. The Survivability Map relates intrusions and the corresponding softspots to specific strategies to improve the resistance, recognition, and recovery capabilities of the system architecture.

Intrusion Scenario	Softspot Effects	Architecture Strategies for →	Resistance	Recognition	Recovery
(Scenario 1)		Current			
...		Recommended			
(Scenario n)		Current			
		Recommended			

Figure 4: Survivability Map Template

As noted, survivability deals with adverse conditions arising from intrusions, failures, or accidents. The SNA method focuses on intrusions and compromises in order to capitalize on both extensive CERT/CC experience with intrusion analysis and the CERT/CC intrusion knowledge base. The SNA method is equally applicable to analysis of failures and accidents, and such analysis is easily incorporated.

5 Survivable Network Analysis Process

5.1 Planning and Conducting an SNA

An SNA is conducted through a series of Joint Sessions by a customer team and an SEI team, as depicted on the left side of Figure 5. The SEI team also performs the series of Analytical Tasks, shown on the right side of Figure 5. The SNA is initiated through a Joint Planning Meeting, and culminates in a Joint Briefing Session that summarizes findings and recommendations. Each step in the SNA process is described below. Joint Session responsibilities are labeled as "Customer," "SEI," or "Joint," as appropriate.

5.2 Joint Planning Meeting

The purpose of the Joint Planning Meeting is to assign responsibilities and make all preparations for the conduct of the SNA.

SEI Responsibilities:

1. Establish the SEI team membership. Teams are typically composed of three members.
2. Assign a team leader who will also serve as the single point of contact (POC).

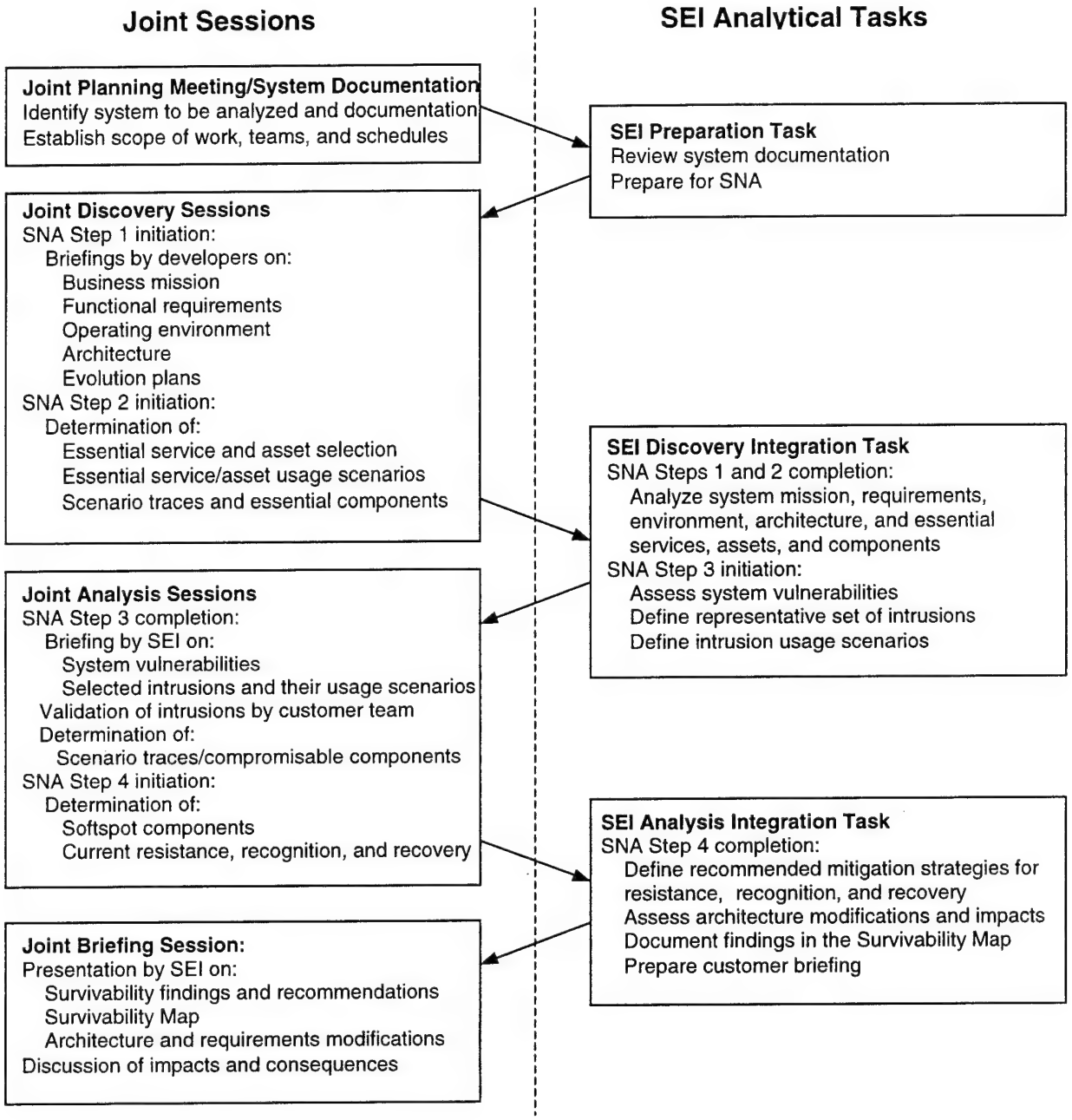


Figure 5: SNA Sessions and Tasks

Customer Responsibilities:

1. Establish the customer team membership. The customer team should include expertise in the system mission, requirements, operating environment, usage, and architecture. Typical team membership might include the system architect, a lead designer, and several stakeholders, including system owners, who have knowledge of how the system addresses business objectives, and system users, who have knowledge of usage needs and patterns. Four to six team members can usually provide sufficient representation.
2. Assign a team leader who will also serve as the single point of contact. The team leader should have the authority to identify appropriate team members and ensure their participation.
3. Identify the system to be analyzed. The system or system part to be analyzed should exhibit the following characteristics:
    - It should be an appropriate size to permit effective analysis within the time and resource constraints of the SNA. Large systems can be productively analyzed at high levels of granularity to produce general findings with broad scope. Small systems can be analyzed more extensively to produce detailed findings with localized scope. The granularity of the analysis will be adjusted to account for system size and complexity.
    - It should exhibit clear boundaries. All system interfaces and connections must be known and understood. For example, every network connection should be known.

Joint Responsibilities:

1. Scope the system to be analyzed and establish bounds for the SNA process.
2. Establish schedules for the work, and venues for the joint sessions.

Exit Criteria:

SEI and customer teams and team leaders are assigned, the system to be analyzed is identified, and schedules are set. Required documentation is also identified as described below.

5.3 System Documentation

The SNA process is facilitated by documentation of the system to be analyzed. The documentation is typically selected from existing materials, and provides the explanatory and reference information required for the analysis. Note that all required documentation may not be available. Even partial documentation is valuable, however, and information gaps can typically be filled during the working sessions.

Customer Responsibilities:

Provide system documentation to the SEI team that describes the following:

- *Business mission*. Define the principal objectives of the system from a business perspective.
- *Functional requirements*. Define the major functions of the system in terms of:
  - principal transactions available to each class of user
  - requirements for information access and retention
  - processing volumes and rates
- *Operating environment and users*. Define the operational characteristics of the system in terms of:
  - locations and environments of system components
  - classes and access capabilities of users, including nominal system users, developers, maintainers, operators, and administrators
  - operating procedures, including system monitoring and control, procedures for control of users, security administration, maintenance methods, and backup and recovery processes
- *Architecture*. Define the system configuration in terms of:
  - hardware components and their connections (typically in block diagram form) including all external access points and communication links
  - software components resident in every type of hardware component, including all protocols, operating systems, application programs, databases, repositories, and security, maintenance, backup, and recovery facilities
  - human components, including all administrators, developers, maintainers, and operators

Exit Criteria:

Required documentation is identified and reviewed.

5.4 SEI Preparation Task

The SEI team will review the system documentation and prepare for the joint discovery sessions.

5.5 Joint Discovery Sessions

Customer responsibilities:

The customer team initiates SNA Step 1, System Definition, by providing briefings on the business mission, principal functional requirements, system architecture, operational environment, typical usage scenarios, and evolution plans for the system. This information provides an understanding of the system for all SNA participants.

Joint Responsibilities:

Both teams initiate SNA Step 2, Essential Capability Definition. The customer identifies a set of essential services and assets whose availability must be maintained in adverse conditions

and the usage scenarios that invoke and access them. Both teams trace the scenarios through the architecture to identify corresponding essential components. Based on resources and schedules available for the SNA, a set of three or four of the highest priority essential services and assets is typically identified.

Exit Criteria:

Both teams share a common level of understanding of the system, essential services and assets have been identified, and the scenarios have been traced through the architecture to reveal the essential components.

5.6 SEI Discovery Integration Task

SEI Responsibilities:

The SEI team completes SNA Steps 1 and 2 by analyzing and summarizing the system mission, functional requirements, operational environment, essential services and assets, scenario traces, and essential components. Based on this information, the team initiates SNA Step 3, Compromisable Capability Definition, by assessing system vulnerabilities, identifying a set of representative intrusions that the system could experience, and defining their corresponding usage scenarios.

Exit Criteria:

System vulnerabilities and representative intrusions have been identified.

5.7 Joint Analysis Sessions

SEI Responsibilities:

The SEI team provides a briefing on the identified vulnerabilities and representative intrusion scenarios.

Customer Responsibilities:

The customer team validates the selected intrusion scenarios, possibly proposing modifications or extensions.

Joint Responsibilities:

Both teams complete SNA Step 3 by tracing the intrusion scenarios through the architecture to reveal the corresponding compromisable components. The teams also initiate SNA Step 4, Survivability Analysis, by identifying the softspot components (both essential and com-

promisable), and proposing and discussing potential resistance, recognition, and recovery strategies.

Exit Criteria:

Representative intrusions have been validated, their scenarios have been traced through the architecture to reveal the compromisable components, and initial mitigation strategies have been discussed.

5.8 SEI Analysis Integration Task

SEI Responsibilities:

The SEI team completes Step 4 by reviewing the results of the Joint Analysis Sessions and developing findings and recommendations for mitigation that address resistance, recognition, and recovery strategies. The strategies are defined as modifications to the current system architecture and are summarized in a Survivability Map. A customer briefing is prepared to review the SNA findings and recommendations.

Exit Criteria:

Recommendations are formulated and the briefing is prepared.

5.9 Joint Briefing Session

The briefing session is conducted by the SEI team to present the findings and recommendations developed during the SNA. The following areas are covered:

- principal business mission, requirements, and operating environment of the system
- current system architecture
- selected essential services and assets and their usage scenarios
- essential system components
- selected intrusions and their usage scenarios
- compromisable system components
- resistance, recognition, and recovery analysis
- recommended architecture modifications and Survivability Map

The briefing session is attended by the customer team and customer management. The findings and recommendations are discussed and next actions are explored.

6 Recent Results

6.1 The SNA Client Report

In our work with SNA clients we have found that a number of different methods and templates are helpful for developing a good set of recommendations. A typical SNA report has the following sections:

- Executive Summary
- Sections
  1. Overview
  2. The Survivable Network Analysis Method
  3. Architecture
  4. Essential Services
  5. Intrusion Scenarios
  6. Recommendations
  7. Implementation
- Appendices, and References

The Executive Summary provides a snapshot of SNA recommendations and a brief discussion of the system under study. The Overview gives a brief introduction to the project. The Survivable Network Analysis Method section gives an abbreviated method description. Architecture describes the architecture of the system under study. Essential Services describes the business processes of the organization, normal usage scenarios, and essential service scenarios and components. Intrusion Scenarios describes general attacker profiles, specific system attack impacts, specific system attacker profiles, intrusion scenarios, attack patterns, representative intrusions, attacker types, and mapping of intrusion scenarios to the essential services. Recommendations are in the areas of policy and architecture and include the Survivability Map. For Implementation, timeline and resource considerations are discussed.

The SNA Method description that appears earlier in this report is usually abbreviated for Section 2 of the SNA client report. In Section 3 we provide a diagram of the architecture. An example of such a diagram is shown in Figure 6.

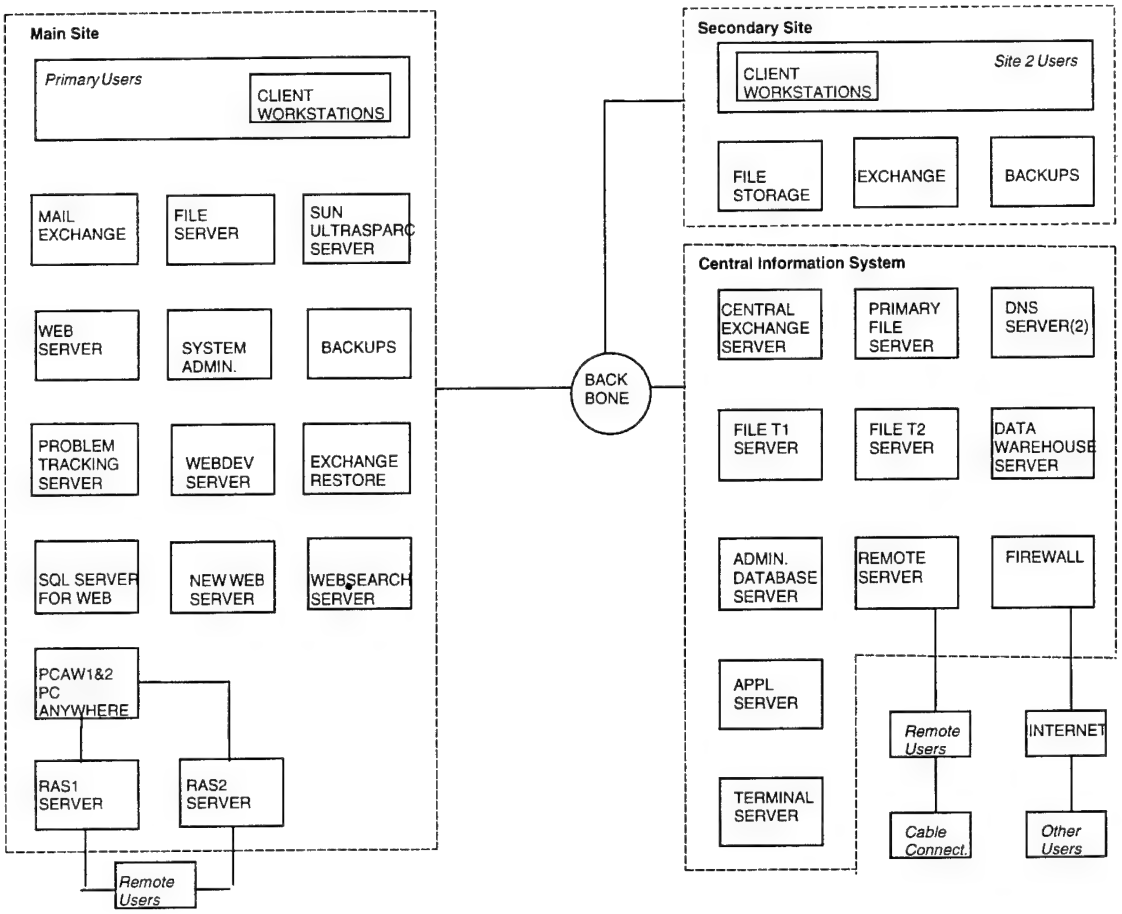


Figure 6: Sample Architecture Diagram

6.2 Essential Services

In Section 4 we provide a mapping of business processes to essential services. A template for this mapping is shown in Table 3.

Table 3: Mapping of Business Processes to Essential Services

Business Processes / Essential Services/Assets	Process 1	Process 2	Process 3
Essential service 1/ Asset 1	X		
Essential service 2/ Asset 2	X	X	
Essential service 3	X	X	X
Essential service 4/ Asset 4		X	X

The architecture diagram is modified to show the service components in bold, for each of the essential-service scenarios. A single diagram shows the union of these diagrams, providing a snapshot of all essential components. Figure 7 illustrates this.

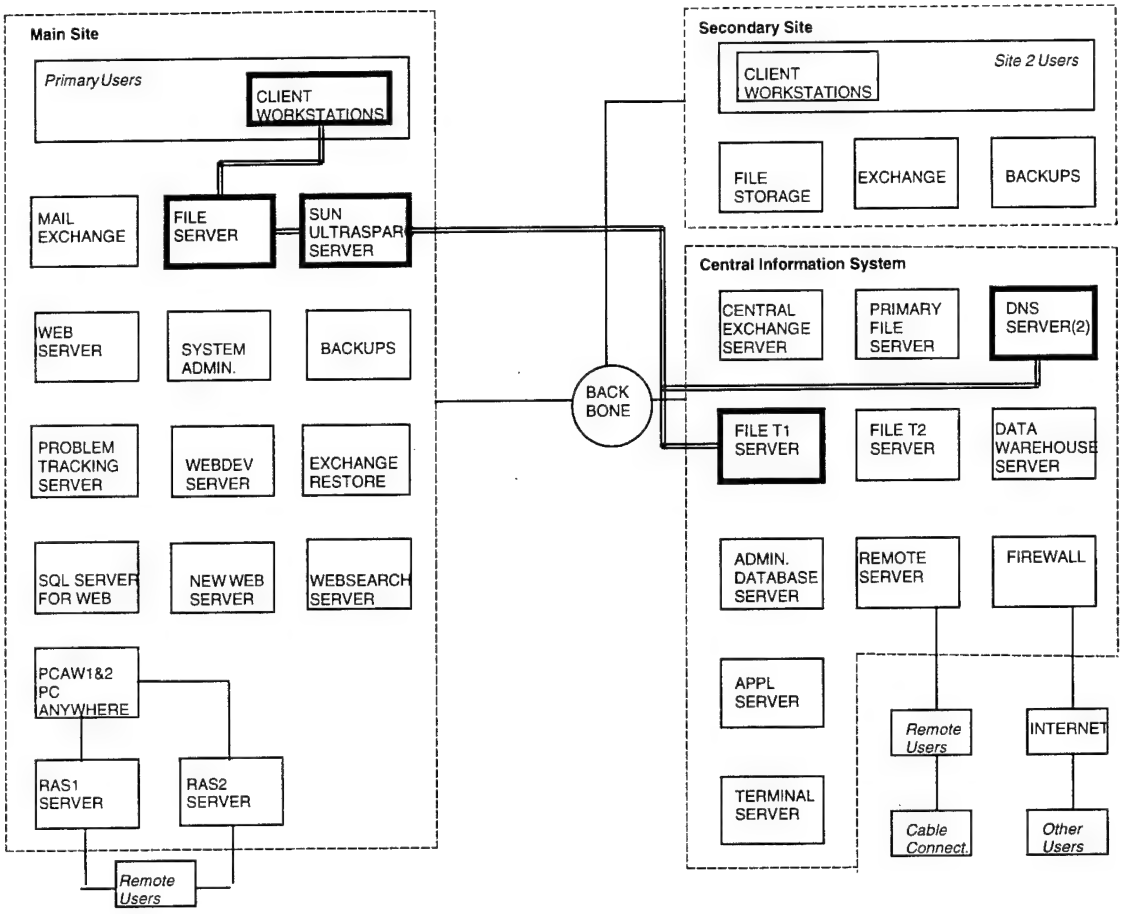


Figure 7: Architecture with Essential Service Components Highlighted

6.3 Intrusion Scenarios

6.3.1 Attacker Attributes

In Section 5, the Intrusion Scenarios section, classes of attackers are identified. For each class of attacker, the following attributes are noted:

Resources — The resources that an attacker can draw upon. Resources include funds, personnel, and the skill levels of those personnel.

Time — An attacker can have very-near-term objectives or can be very patient and wait for an opportunity. A patient attacker is better able to avoid detection by spreading system probes over a longer time and initiating those probes from multiple

locations.

Tools — Many attacks are now supported by tool sets, which means that attackers can be successful with a lower level of skill than was once required. The attacks generated by such tool sets can also have a unique series of steps (i.e., a signature that can be recognized by an intrusion-detection system). The sophisticated attacker can tailor those tools to change the signature and hence to avoid detection, or can develop tools to target a specific system.

Risk — The risks that attackers are willing to bear often depend on their objectives. An activist will want the attack to be publicly known, and terrorists may claim credit for an attack. On the other hand, an attacker who is trying to obtain sensitive industrial information may not want the target to know that an attack even took place.

Access — Intruder access is described in terms of access mechanisms, such as dialup or Internet access, and in terms of system boundaries, such as inside or outside a firewall or LAN.

Objectives — An attacker's objectives include personal gain, such as recognition or improved hacking skills, embarrassment of the target organization, and financial gain.

6.3.2 Levels of Attack

Three levels of attack are considered: target of opportunity, intermediate, and sophisticated attacks.

6.3.2.1 Target-of-Opportunity Attack

The most frequent kind of attack is called a "target-of-opportunity attack" and is typically associated with the recreational attacker. Disgruntled employees with some organizational knowledge and system skills could also fall into this category.

The following apply to target-of-opportunity attacks:

- The attacker has a very general objective and hence a broad range of targets. The immediate impact might be a denial of service, but Web data could also be affected.
- The attacker uses readily available tools to scan and probe systems to take advantage of known vulnerabilities.

- The attacker may not require user access to internal systems—for example, an attack could be launched using an email virus.
- The attacker has limited knowledge of internal systems, processes, and staff.
- There is a high frequency of these attacks.
- The internal machines involved may be used to attack other sites.
- These attacks have a relatively low impact on well-administered systems, but there are exceptions.

The traditional defenses against target-of-opportunity attacks are firewalls to control access, monitoring systems and applications to learn of new vulnerabilities, and regular system upgrades to remove known vulnerabilities.

6.3.2.2 Intermediate Attack

Intermediate attacks typically have an organization-specific objective. Such an attacker will perform the same kinds of scans and probes as would the recreational attacker but can better hide such activity. Existing system vulnerabilities will make the attacker's task much easier. The following are characteristics of the intermediate attack:

- The attack may initially compromise one of the trusted external systems.
- The attacker will likely have considerable patience and skill.
- There is a higher probability of success, compared with a target-of-opportunity attack, and a greater likelihood that essential services will be affected.

6.3.2.3 Sophisticated Attack

The sophisticated attacker has a very specific organizational objective and can significantly affect essential services. This type of attacker may also attempt to compromise internal staff.

Whereas the defense against a target-of-opportunity attack could concentrate on prevention, sophisticated attacks, while very rare, have a high probability of success, and are likely to overcome preventive measures. The most significant issue for an organization that is the target of a sophisticated attack are recovery and the recognition that the system has been compromised.

The following are characteristics of a sophisticated attack:

- The attacker will be very patient.
- The attacker will expend significant time collecting information on the system's architecture and staff.
- The attacker will have a focused objective.
- The attacker can customize or develop tools.

- There is a high probability of success.

6.3.3 Attacker Profiles

Attacker profiles, as shown in Table 4, are also considered in an SNA Report. These attacker profiles can be tailored for a particular client organization.

Table 4: Attacker Profiles

Attacker	Resources	Time	Tools	Risk	Access	Objectives
Recreational Hacker	Range of skills Many have limited ability May operate as part of a team	Can be patient, but usually looks for opportunity	Uses readily available tool sets	May not understand or appreciate the risk	External	Personal recognition Develop hacking skills
Disgruntled Employee	Depends on personal skills May have knowledge of process Unlikely to use external resources	Could be very patient and wait for opportunity	Uses readily available tool sets Former system admin could develop tools	Risk averse particularly if still employed	Internal or external Internet or LAN	Personal gain Embarrass organization
Activist who targets organization for ethical or political reasons	Limited means to hire external expertise, but could have talented members	Likely very patient, but specific events may force quicker action	Uses readily available tool sets	Not risk averse	External Internet	Embarrass organization Impact public or customer opinions Impact government or corporate partners
Industrial Spy	Expert knowledge	Desired information has limited shelf life	Can customize tools	Somewhat risk averse Capture could impact corporate sponsors	External Internet	Sell proprietary information Gain knowledge of competitor's research, learn of corporate strategies

Nation-State	Could hire external resources for high-payoff attack	Patient, but desired information may be needed quickly	Could develop tools if payoff is high	Moderately risk averse and may be able to operate outside of U.S.	External and Internet Could be organizational visitor	Access government information or corporate proprietary information

6.3.4 Attack Patterns

We also consider attack patterns. Attacks fall into three general patterns:

1. *User access.* Attacks in this category require system access with user or system-wide privileges. The steps in this kind of attack are:

   a. **Gather information**. Perform an exhaustive search to gather system data and to identify existing security vulnerabilities. This step is often automated with tools such as nmap[1] or tools that target a specialized application such as a Web server. Vulnerabilities might exist in the system components, can result from errors in system administration, or can be reflected in poor security policies.

   b. **Exploit.** Exploit a security hole to gain access or obtain system information for use in a later attack. In the early stages of an attack, the vulnerability may provide information such as machine names or user account names.

   c. **Damage**. Bring about the desired effect of the attack through such means as changing data, accessing sensitive information, or establishing a permanent connection that can be used for continuing access. The final step in this attack is to attempt to change logs so that the attack is not identified.

2. *Component access.* An attack in this category does not require user access on the system. These attacks create a denial of service by sending improper requests. In some instances, such a request can crash poorly designed system components. In other cases, the extra time that is required to process such a request is sufficient to noticeably slow down processing. The steps in this attack are:

   a. **Gather information**. Identify a systems component and communications port.

   b. **Exploit.** Send messages to the selected port.

   c. **Damage**. Crash or overload an application component or network service.

3. *Application content.* These attacks send improper data to applications rather than to network components. In this situation, the network traffic is properly formatted. The problem is with the content of the traffic, and like the network access attacks these examples do not require that the attacker obtain user access. The steps are:

---

[1] nmap is a freely available tool that scans a network. It is available at http://www.insecure.org/nmap/

a. **Gather information**. Identify the target application. This could be either a network-based application such as a Web server or browser, or an application such as Microsoft Office, where email is used to transmit the data to the application.

b. **Exploit.** Send content directly or indirectly (via, say, email) to the target application.

c. **Damage.** Remove user files, change the configuration of a user workstation, and export user files.

As an example, consider a virus attack that follows the application content pattern. It might occur as follows:

Gather information:

- Identify internal mail aliases or user IDs.
- Identify the mail client software (e.g., Outlook).
- Identify the Web browser.
- Learn which scanners are used to detect mail viruses.
- Learn the internal mail-processing architecture in terms of servers as well as Mail Transfer Agents.

Exploit:

- Attach a damaging Visual Basic macro to a Word or Excel file.
- Trick a user into downloading a virus from a Web site.
- Use a scripting language such as JavaScript to fool a user into using a site, which can capture information exchange.

Damage:

- Remove user files.
- Reconfigure a user workstation to support an attack.
- Install a communications backdoor on a user workstation.
- Capture user passwords or other confidential information.
- Export sensitive information via an automatic mailing of current messages or Word documents.

6.4 Recommendations

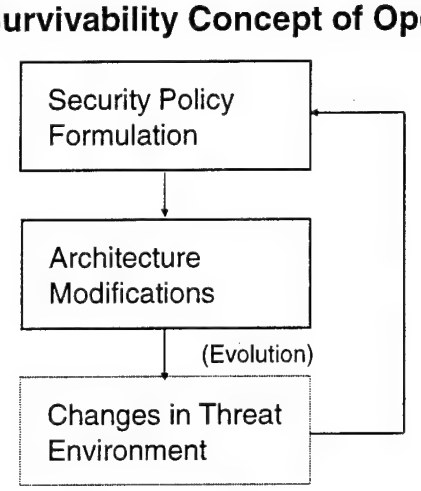


Figure 8: Relationship of Policy, Architecture, and Threat

The recommendations made in Section 6 of a client report typically include a survivability concept of operations, as shown in Figure 8. Our recommendations tend to include both policy and architecture recommendations:

6.4.1 Policy Recommendation

The following is an example of a policy recommendation:

> ***Clarify policy on active content, such as XML.***
>
> *One person's data is another's program. We have seen increased examples of malicious content in the past few years. Incidents such as the Melissa virus call for clear and carefully considered policies in this area. Policy components could range from requiring appropriate screening of all active content that enters an organization to the outright banning of certain forms of active content. The creation and dissemination of active content, such as content using the Extensible Markup Language (XML), Active X controls, Java applets, and JavaScript by both internal and public organizational Web services, should be carefully considered and controlled.*

6.4.2 Architectural Recommendation

A specific architectural recommendation for virus management would have the following form:

Type: Application, Network, Infrastructure (systems)

Strategy: Recognition, Resistance, Recovery

Intrusions: Identify to which of the intrusions scenarios identified for the client the recommendation applies (in the example the scenarios are identified by number).

Rationale: Provide justification and context for recommendation.

Result: Make an explicit recommendation that can be action for the client.

Implementation: Discuss general implementation options, risks, and tradeoffs with other recommendations.

The following is an example of an architectural recommendation:

> ***Examine incoming email Web content and establish virus-management practices.***
>
> **Type:** Application
>
> **Strategy:** Recognition and Recovery
>
> **Intrusions:** 3, 4, 5
>
> **Rationale:** Surveys of attacks have identified email viruses as having a very high financial impact as well as a high frequency of occurrence. Email or Web pages increasingly contain what is called "active content." For email this may be an attachment such as a Word or Excel document that includes macros. Web pages can contain Java or JavaScript code. Viruses such as Melissa make use of attachments that contain damaging macros. A virus attack is most likely a part of a general Internet attack. Your organization could receive such a message in the early phase of an attack before the organization's virus-scanning software has been updated to reflect the latest incident. A sophisticated or intermediate level of attack could create a virus targeted for a specific organization.
>
> **Result:** Establish practices for crisis management with respect to email viruses. Maintain awareness of general risks and the limitations of scanning techniques.
>
> **Implementations:** Implementations to improve recognition could be implemented at multiple levels:
>
> a. Do a virus scan at the mail hub.

b. Scan mail messages on each Microsoft Exchange server.

c. Scan files on the file system or portable PCs (includes viruses that might be transported by Web access).

There should be recommended user configurations for mail clients such as Microsoft Outlook as well as Web browsers used for mail to restrict the use of scripting in mail messages. There should be advice for the user community to address viruses or other active-content vulnerabilities (e.g., ActiveX) associated with Web pages and mail messages.

Of equal importance are improvements for recovery. A virus like Melissa can overwhelm a mail system and create a denial of service for that essential component. Other viruses may remove user files or change configurations on user workstations. Improved recovery could include the following steps:

a. Establish plans to respond to a virus not initially identified by the virus-scanning software.

b. Establish or review plans to restore a significant portion of user files.

c. Establish or review plans to restore a significant number of user workstations

The recommendations also include development of a Survivability Map, which was discussed earlier in this report.

6.5 Implementation, Appendices, and References

Section 7, the Implementation section, includes a timeline and rough estimate of resources. These are shown below in Table 5 and Table 6.

Table 5: Timeline Phasing of Recommendations

Type	Short Term 1-6 months	Mid Term 6-12 months	Long Term 18+ months
Policy	P3: Clarify policy on active content P5: Terminate/change access rights P7: Eliminate internal use data from Web site P8: Review physical security P11: Establish security training procedures P12: Clarify security roles	P1: Clarify access rights P2: Review authentication domains P4: Define access control mechanisms P6: Log/monitor host logs for intrusions P9: Review/exercise backup and recovery P10: Audit C drives of desktops and portables	P13: Develop a security policy
Architecture	R1: Add firewalls to create a DMZ R3: Establish separate servers for internal sensitive data and external public data R5: Eliminate PC Anywhere R6: Add workstation time-outs on sensitive data access R9: Examine incoming email Web content R10: Establish host-based defense on Web servers	R2: Establish access policies based on employee status R4: Use encrypted channels for remote access R8: Monitor integrity of public Web data R13: Separate development and production of Web materials R15: Monitor outgoing network traffic	R7: Protect all data on portables R11: Notify on every update to sensitive data
Operations	Operational procedure implementation of policy and architecture recommendations	Operational procedure implementation of policy and architecture recommendations R12: Shunt attacks as necessary R14: Establish procedure to move Web data from development to production server	Operational procedure implementation of policy and architecture recommendations

Table 6: Estimated Relative Resources to Implement Recommendations

Recommendation	Labor	Equipment
P1:Clarify access rights	Low	Existing
P2: Review authentication domains	Low	Existing
P3: Clarify policy on active content*	Low	Existing
P4: Define access-control mechanisms	Low	Low
P5: Terminate/change access rights*	Low	Existing
P6: Log/monitor host logs for intrusions	Medium	Existing
P7: Eliminate internal use data from Web site*	Low	Existing
P8: Review physical security*	Low	Low
P9: Review/exercise backup and recovery	Medium	Existing
P10: Audit C drives of desktops and portables	Medium	Existing
P11: Establish security training procedures*	High	Existing
P12: Clarify security roles	Medium	Existing
P13: Develop a security policy	High	Existing
R1: Add firewalls to create a DMZ*[2]	High	Medium
R2: Establish access policies based on employee status	Medium	Medium
R3: Establish separate servers for internal sensitive data and external public data*	Low	Medium
R4: Use encrypted channels for remote access	Medium	Low
R5: Eliminate PC Anywhere*	Low	Existing
R6: Add workstation timeouts on sensitive data access*	Low	Existing
R7: Protect all data on portables	Medium	Low
R8: Monitor integrity of public Web data	Medium	Low
R9: Examine incoming email Web content*	High	Low
R10: Establish host-based defense on Web servers*	Medium	Low
R11: Notify on every update to sensitive data	Medium	Existing
R12: Shunt attacks as necessary	Medium	Existing
R13: Separate development and production of Web materials	Low	Low
R14: Establish procedure to move Web data from development to production server	Low	Existing
R15: Monitor outgoing network traffic	Medium	High

Labor: Low = 0 - 1 PM
Medium = 1- 2 PM
High = 2+ PM
PM = Person-Month

Equipment: Existing = current equipment
Low = $ 1 – 2 K
Medium = $ 2 – 10 K
High = $ 10+ K

[2] DMZ (de-militarized zone): a network added between a protected network and an external network to provide an additional layer of security.

* = Short-term recommendation

Appendices and references are optional, and will vary from client to client.

6.6 Lessons Learned

In working with clients we have found some variability in the method. This may occur depending on whether the client is defining requirements, specifying an architecture, or doing a major upgrade to an existing system. SNA is easily tailored for those situations.

We have also found that certain attack scenarios allow the intruder to compromise all assets and services, so that the mapping to the architecture becomes trivial, and all essential components are thus softspots. This leads to more global recommendations.

7 Future Research Plans

The Survivable Network Analysis method has helped organizations to define and implement system improvements to deal with inevitable intrusions and compromises in a proactive manner. As stakeholders depending on systems to carry out organizational missions, users, managers, and technical personnel have benefited from increased focus on survivability issues.

A key next step for SNA evolution is to develop more powerful abstractions and reasoning methods for defining the behavior and structure of large-scale distributed systems. Such results will enable more comprehensive analysis of essential service and intrusion traces while limiting complexity. In addition, improved representations and methods are required for defining intrusions. It is important to move beyond the limitations of natural language, to develop uniform semantics for intrusion usage that permits more rigorous analysis and even computational methods to be applied.

Another fruitful line of research involves developing standardized architectural styles or templates for survivability strategies that can be inserted and composed with system architectures to improve their survivability properties. Such templates can be independently analyzed once and for all to define and document their contribution to system survivability.

The larger context for survivability, system life-cycle models, and their associated activities, will be investigated in order to identify a standard set of life-cycle activities in support of survivability, and to identify those life-cycle activities in which further survivability research is needed. In association with this, standardized metrics for survivability are desirable, if indeed such metrics can be identified.

The authors intend to pursue this research agenda as the next step in SNA evolution.

References

[Anderson 97] R. H. Anderson, A. C. Hearn, and R. O. Hundley, "RAND Studies of Cyberspace Security Issues and the Concept of a U.S. Minimum Essential Information Infrastructure," *Proceedings of the 1997 Information Survivability Workshop,* CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University, 1997.

[Boehm 89] B. W. Boehm, *Software Risk Management,* IEEE Computer Society Press, 1989.

[Ellison 98] Ellison, R.; Linger, R.; Longstaff, T.; Mead, N. *Case Study in Survivable Network System Analysis* (CMU/SEI-98-TR-014, ADA355070). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1998. Available at: http://www.sei.cmu.edu/publications/documents/98.reports/98tr014/98tr014abstract.html

[Ellison 99] R. Ellison, D. Fisher, R. C. Linger, H. F. Lipson, T. Longstaff, and N. R. Mead, *Survivable Network Systems: An Emerging Discipline* (CMU/SEI-97-TR-013). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1997, revised 1999. Available at: http://www.sei.cmu.edu/pub/documents/97.reports/pdf/97tr013.pdf

[Kazman 98] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. F. Lipson, and S. J. Carriere, "The Architecture Tradeoff Analysis Method," *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems*, August 1998, Monterey, CA, IEEE Computer Society. Available at: http://www.sei.cmu.edu/ata/

[Marmor-Squires 88] A.B. Marmor-Squires and P.A. Rougeau, " Issues in Process Models and Integrated Environments for Trusted Systems Development", *Proceedings of the 11th National Computer Security Conference*. October 1988.

[Marmor-Squires 89] Ann Marmor-Squires, John McHugh, Martha Branstad, Bonnie Danner, Lou Nagy, Pat Rougeau, and Dan Sterne, "A Risk Driven Process Model for the Development of Trusted Systems," *Proceedings of the 1989 Computer Security Applications Conference*. Tucson, AZ, December 1989. PP 184-192

[Mills 86] H. D. Mills, R. C. Linger, and A. R. Hevner, *Principles of Information Systems Analysis and Design*, New York, NY, Academic Press, 1986.

[Parnas 86] D. L. Parnas and P. C. Clements, A Rational Design Process: How and Why to Fake It. IEEE Transactions on Software Engineering SE-12(2): 251-257, February 1986.

[Royce 87] W. W. Royce, "Managing the Development of Large Software Systems," *Proceedings of the 9th International Conference on Software Engineering,* Monterey, CA, IEEE Computer Society Press, Los Alamitos, CA, 1987.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)		Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Survivable Network Analysis Method	F19628-00-C-0003

6. AUTHOR(S)

Nancy R. Mead, Robert J. Ellison, Richard C. Linger, Thomas Longstaff, John McHugh

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2000-TR-013

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-2000-TR-013

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Society is increasingly dependent on large-scale, networked information systems of remarkable scope and complexity. This dependency magnifies the far-reaching consequences of system damage from attacks and intrusions. Yet no amount of security can guarantee that systems will not be penetrated. Incorporating survivability capabilities into an organization's systems can mitigate the risks. Survivability is the capability of a system to fulfill its mission in a timely manner despite intrusions, failures, or accidents. The three tenets of survivability are (1)resistance to intrusions, (2) recognition of intrusion effects, and (3) recovery of services despite successful intrusions. The survivability of existing or planned systems can be analyzed at the level of system architectures or requirements. This report describes the Survivable Network Analysis (SNA) method developed at the SEI's CERT® Coordination Center. The four-step SNA method guides stakeholders through an analysis process intended to improve system survivability when a system is threatened. The method focuses on preservation of essential system services that support the organizational mission. SNA findings are summarized in a Survivability Map that enumerates current and recommended architectural strategies. SNA has been successfully applied to commercial and governmental systems, and continues to evolve toward increasing rigor in its application.

14. SUBJECT TERMS	15. NUMBER OF PAGES
survivable systems, information security, architecture, analysis	58

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18 298-102